REMARKS:

In accordance with the foregoing, claims 1, 3-9, 17-32 and 45-51 have been amended for clarification, and claims 2 and 10 are cancelled without prejudice. New claim 57 is added. No new matter has been added. Thus, claims 1, 3-9 and 11-57 are pending and under consideration.

REJECTION UNDER 35 U.S.C. §103(a):

In the outstanding Office Action, claims 1-16 were rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 5,778,219 ('219) in view of U.S. Patent No. 5,920,710 ('710). Claims 19-30, 33-44, 46-50, 52 and 53 were rejected under 35 U.S.C. §103(a) as being unpatentable over '219 in view of U.S. Patent No. 5,920,710 ('710).

At pages 6 and 7 of the outstanding Office Action, the Examiner also rejects claims 19-30, 33-44, 46-50, 52 and 53 under 35 U.S.C. §103(a) as being unpatentable over '219 as applied to independent claims 17, 18, 31, 32, 45 and 51, and further in view of '710. The rejections are traversed below and reconsideration is respectfully requested.

'219 discusses a method for supporting speculative operations where exceptions generated by the speculative operations are differed while exceptions by non-speculative operations are immediately reported.

'710 discusses a method for modifying status bits in a reorder buffer with a large speculative state where instructions subsequent to a mispredicted branch are discarded in an event of a mispredicted branch in preceding speculative instructions.

The present invention discloses a processor and method of controlling the same where a load instruction arranged after a store instruction in a program is executed prior to the execution of the store instruction.

The Examiner compares the '219 method for deferring exceptions generated by speculative operations to avoid unnecessary exception processing with the present invention. The '219 method determines speculative and non-speculative versions of operations in an instruction of a CPU (see, column 8, lines 14-18 and column 8, lines 31-39 of '219). If a non-speculative operation generates an error, the '219 functional unit reports the error immediately, and if a speculative instruction generates an exception, the '219 functional unit will defer reporting the exception (see, column 8, lines 22-28 of '219). Accordingly, the '219 system defers exceptions generated by speculative operations until the result of the operation is actually used in the execution path of a program (see, column 10, lines 5-8 of '219). This means that the

'219 method is limited to deferring an error of an operation that causes a program to branch to a different routine unless the operation is actually used in the program.

As recited in amended independent claims 1 and 9, the present invention includes "executing a second instruction that is placed after a first instruction in the program prior to execution of the first instruction" where "the first instruction is a store instruction to store the first data into a storage unit, and the second instruction is a load instruction to read out the second data from the storage unit". This allows the load instruction (the second instruction) to be executed prior to the store instruction (the first instruction), thereby preventing a problem caused by ambiguous memory reference where data stored prior to a store operation is read out by a load operation causing change in execution sequence of instructions. The '219 method does not teach or suggest executing "a load instruction" prior to executing "a store instruction" as disclosed by the present invention.

The Examiner acknowledges that the '219 method does not expressly discuss overwriting an execution result of a first instruction data corresponding to an address of the first data, thus relies on '710 as teaching the same. The '710 microprocessor employs branch prediction in order to speculatively fetch instructions subsequent to conditional branch instructions, where functional units execute the branch instructions and determine if the predicted branch direction is incorrect (see, column 7, lines 43-45 and lines 62-64 of '719). When a mispredicted branch instruction is detected, instructions *subsequent* to the mispredicted branch are discarded from the microprocessor (see, column 8, lines 2-6 and column 11, lines 6-19 of '719). This means that the '719 method is directed to removing speculative instructions upon an occurrence of a mispredicted branch when the mispredicted branch precedes the speculative instructions.

In contrast, independent claims 1 and 9 of the present invention recite, "overwriting an execution result of the first instruction ... when the address of first data to be executed by the first instruction is included in an address region of second data to be processed by the second instruction". This means that when address regions of first and second data interfere, the execution result of a store instruction (first instruction) to store the first data into a storage unit is overwritten, where the first instruction precedes the second instruction. The '719 method does not teach or suggest "overwriting an execution result of the first instruction [preceding the second instruction]... when the address of first data to be executed by the first instruction is included in an address region of second data to be processed by the second instruction".

The combination of '219 and '710 results in a method of supporting speculative

operations according to which exceptions generated by the speculative operations are differed while exceptions by non-speculative operations are immediately reported, and where instructions subsequent to a mispredicted branch are discarded in an event of a mispredicted branch in preceding speculative instructions.

It is submitted that the independent claims are patentable over the combination of '219 and '710.

For at least the above-mentioned reasons, claims depending from independent claims 1 and 9 are patentably distinguishable over the combination of '219 and '710. The dependent claims are also independently patentable. For example, as recited in claims 8 and 14, overwriting is carried out "in accordance with a program at a branch destination designated by executing a branch instruction, when the address of the first data to be processed by the first instruction is included in the address region of the second data to be processed by the second instruction". The combination of '219 and '710 does not teach or suggest "overwriting" a store instruction (first instruction) "in accordance with a program at a branch destination designated by executing a branch instruction when the address of the first data to be processed by the first instruction is included in the address region of the second data to be processed by the second instruction".

Therefore, withdrawal of the rejection is respectfully requested.

REJECTION UNDER 35 U.S.C. §102(b):

In item 12 of the outstanding Office Action, claims 17, 18, 31, 32, 45 and 51 were rejected under 35 U.S.C. §102(b) as being anticipated by '219. The rejection is traversed below and reconsideration is respectfully requested.

Independent claims 17, 18, 31 and 32 of the present invention recite, "executing an instruction placed after a branch instruction prior to execution of the branch instruction" and "retaining an exception operation when necessity of the exception operation is detected in the execution". The exception is performed "when the retained exception operation is needed in execution of an instruction at a branch destination selected through the execution of the branch instruction by a commit instruction" where a program returns to sequentially execute the instructions "starting from the exception start instruction" (claims 18 and 32) or "continue the instruction at the branch destination" (claims 17 and 31) when the exception operation is finished. This is unlike the '219 method where an error tag bit of an operand defers processing of an exception and determines whether the processor will interpret the data field as correct data

or an exception (see, column 14, lines 34-47 of '219). Thus, the '219 method does not teach or suggest "retaining an exception operation when necessity of the exception operation is detected in the execution" and performing the exception "when the retained exception operation is needed in execution of an instruction at a branch destination selected through the execution of the branch instruction by a commit instruction".

Independent claims 45 and 51 of the present invention recite, "retaining a break operation when necessity to suspend execution of the program is detected in executing the instruction by an exception inhibiting load instruction", where the break operation is performed "when the retained break operation is required in execution of an instruction at a branch destination through the execution of the branch instruction". The '219 method does not teach or suggest determining or detecting the need to suspend execution of a program using "an exception inhibiting load instruction".

It is submitted that the independent claims 17, 18, 31, 32, 45 and 51are patentable over '219.

The above arguments discussing the combination of '219 and '710 is hereby incorporated to address the rejection of dependent claims 19-30, 33-44, 46-50, 52 and 53.

For at least the above-mentioned reasons, claims depending from independent claims 17, 18, 31, 32, 45 and 51 are patentably distinguishable over the combination of '219 and '710. The dependent claims are also independently patentable. For example, as recited in claims 38, 44, the processor of the present invention includes, "an identification information rewrite unit that executes a predetermined instruction so as to rewrite the identification information". The combination of '219 and '710 does not teach or suggest, "executing a predetermined instruction so as to rewrite the identification information" that has been read out. Thus, dependent claims 19-30, 33-44, 46-50, 52 and 53 are patentably distinct over the combination of '219 and '710.

Therefore, withdrawal of the rejection is respectfully requested.

NEW CLAIM:

New claim 57 is added to emphasize a method of the present invention that includes, "executing a load instruction located after a store instruction in a program sequence prior to execution of the store instruction" where it is detected "...whether an address region of data to be executed by the store instruction overlaps with an address region of data executed by the load instruction" and "an execution result of the store instruction is overwritten upon determining that the address region for the execution of the store instruction overlaps the address region of

the data executed by the load instruction". The cited references in combination or alone do not teach or suggest "detecting whether an address region of data to be executed by the store instruction overlaps with an address region of data executed by the load instruction" and "overwriting an execution result of the store instruction upon determining that the address region for the execution of the store instruction overlaps the address region of the data executed by the load instruction".

Therefore, new claim 57 is patentably distinguishable over the cited references.

CONCLUSION:

There being no further outstanding objections or rejections, it is submitted that the application is in condition for allowance. An early action to that effect is courteously solicited.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 8/13/4

By: _____

J. Randall Beckers
Registration No. 30,358

1201 New York Avenue, NW, Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501